Towards Convergence of Learning Classifier Systems Value Iteration

Jan Drugowitsch and **Alwyn M Barry**[1]

Abstract. In this paper we are extending previous work on analysing Learning Classifier Systems (LCS) in the reinforcement learning framework [4] to deepen the theoretical analysis of Value Iteration with LCS function approximation. After introducing the formal framework and some mathematical preliminaries we demonstrate convergence of the algorithm for fixed classifier mixing weights, and show that if the weights are not fixed, the choice of the mixing function is significant. Furthermore, we discuss accuracy-based mixing and outline a proof that shows convergence of LCS Value Iteration with an accuracy-based classifier mixing. This work is a significant step towards convergence of accuracy-based LCS that use Q-Learning as the reinforcement learning component.

1 INTRODUCTION

The work in [4] describes how to model Learning Classifier Systems (LCS) in the reinforcement learning framework. Even though it is restricted to constant classifier populations, it is a crucial milestone towards a unified theory of function approximation, reinforcement learning and classifier replacement in the context of LCS. In this paper we investigate some properties of the effects of using LCS function approximation in combination with Value Iteration, particularly w.r.t. convergence of the algorithm.

Learning Classifier Systems are a framework for solving Dynamic Programming problems via a rule-based system, where the rules are evolved using a steady-state niche-based GA. The question of convergence of LCS Value Iteration is an important one, as Value Iteration is a deterministic iteration that Q-Learning stochastically approximates. XCSF [8], a derivate of the currently most used classifier system XCS, uses Q-Learning as its reinforcement learning component and is therefore directly affected by our investigations. Even though we could attempt to model Q-Learning in LCS directly, it is more appropriate to first handle the deterministic case and then show that the stochastic approximation is appropriate in the sense of it converging to the deterministic iteration at infinity.

After describing the formal framework of reinforcement learning, LCS function approximation and LCS Value Iteration, we will discuss some properties of vector fields that will guide subsequent convergence investigations. These properties are then demonstrated to apply to the Value Iteration operator as well as single classifiers. With a similar approach we will investigate LCS Value Iteration with i) constant mixing weights, ii) arbitrary mixing, and iii) accuracy-based mixing, where we can prove convergence of the first case and (based on a certain conjecture) the third case, and give an example why the system might not converge for the second case.

[1] Department of Computer Science, University of Bath, Bath BA2 7AY, UK, email: {J.Drugowitsch,A.M.Barry}@bath.ac.uk

2 LCS VALUE ITERATION

This section gives the formal basis of reinforcement learning and Value Iteration and shows how Value Iteration can be applied in LCS.

2.1 The Reinforcement Learning Framework

Let S be the finite set of states of size $N = |S|$, which we will map without loss of generality to the set of natural numbers $\mathbb{N}$. In every state $i \in S$ we can perform an action a from a set of actions A, leading to a transition to the next state $j \in S$ and a scalar reward. The probability of a transition from i to j by performing action a is given by $p_{ij}(a)$, which is the transition function $p : S \times S \times A \to [0, 1]$. Every such transition is mediated by the reward $r_{ij}(a)$, given by the reward function $r : S \times S \times A \to \mathbb{R}$. A policy $\mu : S \to A$ gives the behaviour of an agent, as it determines the action choice for every state. The aim is to find the policy that maximises the discounted reward in the long run; that is for state i, $V^*(i) = \lim_{n\to\infty} \mathbb{E}\left(\sum_{t=0}^{n} \gamma_t r_{i_t i_{t+1}}(a_t) | i_0 = i, a_t = \mu^*(i_t)\right)$, where $\gamma \in (0, 1]$ is the discount factor, and we assume the sequence of states $\{i_0, i_1, \dots\}$ and actions $\{a_0, a_1, \dots\}$ to be generated according to the optimal policy μ^*. $V^* : S \to \mathbb{R}$ denotes the optimal value function that returns the expected return for every state i, and gives the optimal policy μ^* by choosing the action that maximises the expected value of the next state.

2.2 (Approximate) Value Iteration

One way to find the optimal value function V^* is to solve Bellman's Equation

$$V^*(i) = \max_{a\in A} \sum_{j\in S} p_{ij}(a) \left(r_{ij}(a) + \gamma V^*(j)\right), i = 1, \dots, N, \quad (1)$$

which relates the optimal value of a state to the maximum possible reward and discounted optimal value of the next state.

Value Iteration is a method for finding the optimal value function by repeatedly applying the Dynamic Programming (DP) update T to a value vector $V \in \mathbb{R}^N$, holding the current value for every state in S. The update T applied to V is defined by (e.g. [2, Ch. 2.2.1])

$$(TV)(i) = \max_{a\in A} \sum_{j\in S} p_{ij}(a) \left(r_{ij}(a) + \gamma V(j)\right), \quad i = 1, \dots, N.$$

That gives the iteration $V_{t+1} = TV_t$ starting with some arbitrary initial $V_{-1} \in \mathbb{R}^N$. Due to the properties of T, this iteration is guaranteed to converge to the optimal value function V^* when it is applied an infinite number of times.

Given that the set of states is large, calculating the value for every state at every iteration is spatially and computationally prohibitive. Applying function approximation to the value function V is an approach to circumvent this problem. Let $\tilde{V} : S \to \mathbb{R}$ be a parametric function approximation of V that is completely determined by a small set of scalar parameters. The aim at every iteration becomes to minimise the difference between the update according to Value Iteration and its current approximation, that is $\tilde{V}_{t+1} = \min \|T\tilde{V}_t - \tilde{V}\|$, where the minimum is restricted to the approximation space given by the approximation architecture. As discussed in [6, 4], this iteration is only guaranteed to converge for certain function approximation architectures, and might even diverge when used with methods like linear regression or neural networks. Therefore it is important to investigate whether it is compatible with the function approximation in use.

2.3 LCS Function Approximation

Learning Classifier Systems use a special type of function approximation by mixing the independent approximation of a finite set of classifiers to form the overall approximation. Let us consider a set of K classifiers, each identified by its index $k \in \{1, \dots, K\}$. Each classifier k *matches* a certain *matched states set*[2] $S_k \subseteq S$. The objective of each classifier is to minimise the approximation error over its matched states. To ease notation, let $I_{S_k} : S \to \{0, 1\}$ be the indicator function for S_k that returns $I_{S_k}(i) = 1$ if $i \in S_k$ and $I_{S_k}(i) = 0$ otherwise. Depending on the context, we will use I_{S_k} also as a diagonal $N \times N$ *matching matrix*, given by $I_{S_k} = \text{diag}(I_{S_k}(1), \dots, I_{S_k}(N))$. The approximation architecture of a single classifier is linear, based on a set of scalar features that characterise a state, and given by the set of L basis functions $\{\phi_l : S \to \mathbb{R}\}_{l \in \{1,\dots,L\}}$. Together they form the *feature vector* $\phi : S \to \mathbb{R}^L$ that for state $i \in S$ is given by $\phi(i) = (\phi_1(i), \dots, \phi_L(i))'$. The classifier approximation of classifier k is parameterised by the weight vector $w_k \in \mathbb{R}^L$ that gives the value approximation for state $i \in S$ by $\tilde{V}_k(i) = w_k'\phi(i)$. If we combine the state feature vectors into an $N \times L$ feature matrix Φ with $\phi(i)'$ as it's ith row, then we can express the classifier approximation as a vector $\tilde{V}_k \in \mathbb{R}^N$, given by $\tilde{V}_k = \Phi w_k$.

To account for a non-uniform state sampling distribution (determined by the current policy), we will consider the function $\pi : S \to [0, 1]$ to represent the probability of sampling a particular state. That defines the diagonal $N \times N$ diagonal matrix $D = \text{diag}(\pi(1), \dots \pi(N))$, and the equally-sized diagonal matrix $D_k = I_{S_k} D$ for classifier k. For that classifier we want to minimise the mean squared error (MSE) over all states in S_k, weighted by the sampling distribution, and given by[3]

$$\sum_{i \in S} I_{S_k}(i)\pi(i) \left(V(i) - w_k'\phi(i)\right)^2 = \|V - \Phi w_k\|^2_{D_k},$$

where V in the second term is the value function in vector notation. As known from linear algebra, the minimum of the MSE is given by the point of the approximation space of classifier k that is closest to the value vector V. We get this point by orthogonal projection $\Pi_{D_k} V$ into the classifier's approximation space[4] $\{\sqrt{D_k}\Phi w_k : w_k \in \mathbb{R}^L\}$, where Π_{D_k} is the $N \times N$ projection matrix for classifier k, given by $\Pi_{D_k} = \Phi(\Phi' D_k \Phi)^{-1}\Phi' D_k$. For any value function V that gives the optimal approximation of classifier k by $\tilde{V}_k = \Pi_{D_k} V$, with an approximation error of $\|V - \Pi_{D_k} V\|^2_{D_k}$.

Having described the approximation of one classifier, we will now discuss how the classifiers are mixed to give the overall approximation $\tilde{V}$. Let $\psi_k : S \to [0, 1]$ be the mixing weights for classifier k, satisfying $\psi_k(i) = I_{S_k}(i)\psi_k(i)$ and $\sum_{k=1}^K \psi_k(i) = 1$ for all $i \in S$. Let Ψ_k be the $N \times N$ non-negative diagonal mixing matrix $\Psi_k = \text{diag}(\psi_k(1), \dots \psi_k(N))$. Due to the properties of ψ_k, we have $\sum_{k=1}^K \Psi_k = I$, and $\Psi_k = I_{S_k}\Psi_k$ [5]. The overall approximation $\tilde{V}$ given the classifier's approximations $\{\tilde{V}_1, \dots \tilde{V}_K\}$ is then defined by $\tilde{V} = \sum_{k=1}^K \Psi_k \tilde{V}_k$. Hence, for each state it is given by the weighted average of all classifiers that match that state.

As derived in [3], mixing weights that under some assumptions conform to the Maximum Likelihood Estimate are

$$\psi_k(i) = \frac{I_{S_k}(i)\varepsilon_k^{-\nu}}{\sum_{p=1}^K I_{S_p}(i)\varepsilon_k^{-\nu}}, \tag{2}$$

where ε_k is an estimate of the approximation error of classifier k, and $\nu \in \mathbb{R}^+$ is a mixing parameter that is usually set to $\nu = 1$. Hence, the classifiers are weighted inversely proportional to the quality of their approximation. As the approximation error estimate depends on the function to approximate and might change over time, the mixing weights will also change over time, which we will account for by denoting them by $\Psi_{k,t}$.

2.4 LCS Value Iteration

As described before, approximate Value Iteration is based on approximating each step of a Value Iteration. Each classifier maintains approximation $\tilde{V}_{k,t}$ at time t, which gives the overall approximation

$$\tilde{V}_t = \sum_{k=1}^K \Psi_{k,t}\tilde{V}_{k,t}.$$

On this approximation we perform one DP update, giving the non-approximated new value vector $V_{t+1} = T\tilde{V}_t$. At that point we want each classifier to approximate that value vector by minimising $\|V_{t+1} - \tilde{V}_k\|_{D_k}$. This minimum is given by

$$\tilde{V}_{k,t+1} = \Pi_{D_k} V_{t+1} = \Pi_{D_k} T\tilde{V}_t, \quad k = 1, \dots, K.$$

At the same time, each classifier keeps track of the approximation error, which at time $t + 1$ is given by[6]

$$\varepsilon_{k,t+1} = \text{Tr}(D_k)^{-1} \|T\tilde{V}_t - \Pi_{D_k} T\tilde{V}_t\|^2_{D_k},$$

showing that the only time-variant value we need to consider is the current overall approximation[7] $\tilde{V}_t$. As classifier mixing is usually calculated from the classifier errors, the mixing weights $\Psi_{k,t+1}$ are subsequently also a function of $\tilde{V}_t$.

---

[2] Classifiers usually match feature vectors rather than states (and their matching capabilities are restricted by the used representation), but we will assume a bijective map between them, and can therefore treat them as being exchangeable.

[3] $\|\cdot\|_{D_k}$ denotes the weighted norm that is for any vector $z \in \mathbb{R}^N$ given by $\|z\|^2_{D_k} = \sum_{i \in S} I_{S_k}(i)\pi(i)z(i)^2$, that is, weighted by the diagonal of the matrix D_k.

[4] Here we use the fact that for any $z \in \mathbb{R}^N$, $\|z\|_{D_k} = \|\sqrt{D_k}z\|$

[5] Such a set of mixing matrices is called *admissible*.

[6] The error is normalised by $\text{Tr}(D_k)^{-1}$ to make classifiers with different matched state sets comparable.

[7] Note that $\tilde{V}_t$ is usually not explicitly represented but is recovered from the classifier approximations $\tilde{V}_{k,t}$. That requires knowledge of the mixing weights $\Psi_{k,t}$ that are computed from the classifier errors $\varepsilon_{k,t}$. As we use $\tilde{V}_t$ to calculate the next errors $\varepsilon_{k,t+1}$, we need to create a temporary copy of the current errors $\varepsilon_{k,t}$ to be able to calculate $\tilde{V}_t$.

Combining all of the above steps into one update equation gives the LCS Value Iteration update

$$\tilde{V}_{t+1} = \sum_{k=1}^{K} \Psi_{k,t+1} \Pi_{D_k} T \tilde{V}_t. \quad (3)$$

3 CONVERGENCE CONSIDERATIONS

We will describe some of the investigations that we can make regarding convergence of LCS Value Iteration when using averaging classifiers. For a more detailed discussion on the function approximation of averaging classifiers see [3]. The reasons why we restrict ourselves to averaging classifiers is given in [4, Sec. 4.2.1], but can be summarised by the possibility of divergence for other kinds of classifier approximation architectures.

3.1 Vector Field Properties

The algorithm is based on a mapping from $\mathbb{R}^N$ into $\mathbb{R}^N$ that describes an N-dimensional vector field. We will investigate if this vector field forms a contraction[8], which would allow us to use the following property of contraction mappings to show convergence (e.g. [7, Ch. 9.3]):

Theorem 3.1 (Contraction Mapping Theorem)**.** *Let P be a closed real interval, that is P has one of the following forms: $[a, b]$, $[a, \infty)$, $(-\infty, b]$ or $(-\infty, \infty)$. Let $f : P \to P$ be a contraction mapping with contraction modulus $C \in (0, 1)$. Then*

1. *f has a unique fixed point s in P;*
2. *for any $x_0 \in P$, the simple iteration $x_{t+1} = f(x_t)$ gives a sequence converging to s.*

The metric space (M, d) we are operating in is given by $M = \mathbb{R}^N$. We will define the distance metric d as being the maximum norm, given for any two vectors $x, y \in \mathbb{R}^N$ by $d(x, y) = \|x - y\|_\infty = \max_{i=1,\dots,N} |x(i) - y(i)|$.

For the following definitions we are required to order our vectors in some way, which we will for any two vectors $x, y \in \mathbb{R}^N$ define by

$$x \leq y \quad \Leftrightarrow \quad x(i) \leq y(i), \quad i = 1, \dots, N.$$

Definition 3.1 (Increasing Vector Field)**.** Let f be a vector field from $f : \mathbb{R}^N \to \mathbb{R}^N$. Then f is *increasing*, if and only if $x \leq y$ implies that $f(x) \leq f(y)$ for all x and y in $\mathbb{R}^N$.

Definition 3.2 (Scalar Shift Vector Field)**.** Let f be a vector field from $f : \mathbb{R}^N \to \mathbb{R}^N$, let $\gamma \in \mathbb{R}$ be a scalar such that $\gamma \in [0, 1]$, let $m \in \mathbb{R}$ be a scalar, and let $e \in \mathbb{R}^N$ be a vector that is given by $e = (1, \dots, 1)'$. Then we call f a *scalar shift vector field* with scaling γ, if and only if $f(x + me) = f(x) + \gamma me$ for all x in $\mathbb{R}^N$.

If we have a vector field that is increasing and a scalar shift vector field then we can use the following:

Lemma 3.2. *Let $f : \mathbb{R}^N \to \mathbb{R}^N$ describe a vector field that is both increasing and a scalar shift vector field with scaling γ. Then f describes a non-expansion[9] w.r.t. the maximum norm if $\gamma = 1$, and a contraction w.r.t. the maximum norm with contraction modulus γ otherwise.*

Proof. The proof is similar to the one showing the contraction of the DP update operator T in [2, Lemma 2.5]. Let $x, y \in \mathbb{R}^N$ be two vectors, and c be the maximum norm of $x - y$, that is $c = \max_{i=1,\dots,N} |x(i) - y(i)|$. Then we have

$$x(i) - c \leq y(i) \leq x(i) + c, \quad i = 1, \dots, N.$$

Applying f, we can write, based on f's properties,

$$(f(x))(i) - \gamma c \leq (f(y))(i) \leq (f(x))(i) + \gamma c, \quad i = 1, \dots, N.$$

Therefore,

$$|(f(x))(i) - (f(y))(i)| \leq \gamma c, \quad i = 1, \dots, N.$$

Hence we have $\|f(x) - f(y)\|_\infty \leq \gamma \|x - y\|_\infty$ which for $\gamma = 1$ is a non-expansion, and for $\gamma < 1$ is a contraction with modulus γ. □

Therefore, an update function that gives an increasing and scalar shift vector field with scaling < 1 causes convergence if applied iteratively. We will proceed by showing that these properties hold for the DP update T, and then investigate if we can state the same for the DP update in combination with LCS function approximation using averaging classifiers.

3.2 The DP Update T

As the operator T forms the core of Value Iteration, we will discuss some of its properties. The DP update operator T maps from $\mathbb{R}^N$ to $\mathbb{R}^N$ and hence describes a vector field. A simple analysis (as given in [2, Sec. 2.3]) of this field reveals the following properties:

Lemma 3.3. *The vector field given by T is increasing and a scalar shift vector field with scaling γ. Hence, it is a contraction to the maximum norm with contraction modulus γ.*

Proof. The proof for the increasing property and scalar shift property of T is given in [2, Lemma 2.1] and [2, Lemma 2.2]. Its contraction follows from Lemma 3.2. □

Hence, the iteration $V_{t+1} = TV_t$ will converge to the solution of Bellman's Equation $V^* = TV^*$ (Eq. (1)), independent of the initial V_{-1}.

3.3 Averaging Classifiers

Averaging Classifiers are classifiers that use the single feature $\phi(1) = 1$ for their approximation. This results in a $N \times 1$ feature matrix $\Phi = (1, \dots, 1)'$. For the projection Π_{D_k} of classifier k this gives $\Pi_{D_k} = \text{Tr}(D_k)^{-1} \Phi \Phi' D_k$. Applying that to some vector $V \in \mathbb{R}^N$ we get the approximation

$$(\Pi_{D_k} V)(i) = \frac{\sum_{j \in S_k} \pi(j) V(j)}{\sum_{m \in S_k} \pi(m)}, \quad i = 1, \dots, N, \quad (4)$$

which is the distribution-weighted average of V over the matched states S_k. Like T, Π_{D_k} also describes a vector field $\Pi_{D_k} : \mathbb{R}^N \to \mathbb{R}^N$. Some helpful properties of this vector field are:

[8] A function f is a contraction with contraction modulus C if and only if this function is Lipschitz continuous with a Lipschitz constant of $C < 1$. A function f on the metric space (M, d) is Lipschitz continuous if and only if for some non-negative constant $C \in \mathbb{R}$, $d(f(x), f(y)) \leq C d(x, y)$, for all x and y in M. The constant C is called the Lipschitz constant of that function (e.g. [7, Ch. 9.3])

[9] A function f is a non-expansion if and only if it is Lipschitz continuous with a Lipschitz constant $C \leq 1$.

Lemma 3.4. *The vector field described by Π_{D_k} is increasing.*

Proof. Let $V, \bar{V} \in \mathbb{R}^N$ be two vectors such that $V \leq \bar{V}$, and let us denote their non-negative difference by $c = \bar{V} - V$. Using $V = \bar{V} - c$, we get $\Pi_{D_k}\bar{V} = \Pi_{D_k}V + \Pi_{D_k}c$. By substituting Eq. 4 for $\Pi_{D_k}c$ we can see that this term is non-negative in all its components and therefore $\Pi_{D_k}V \leq \Pi_{D_k}\bar{V}$. □

Lemma 3.5. *The operator Π_{D_k} describes a scalar shift vector field with scaling 1.*

Proof. The result follows from expanding for $(\Pi_{D_k}(V+me))(i)$ for an arbitrary vector $V \in \mathbb{R}^N$, state $i \in \{1, \ldots, N\}$, scalar $m \in \mathbb{R}$, and vector $e \in \mathbb{R}^N$ given by $e = (1, \ldots, 1)'$. □

That would be enough to show convergence of LCS Value Iteration with a single classifier. However, we are more interested in the mixed combination of several classifiers and will therefore investigate three cases: i) constant mixing, ii) arbitrary mixing, and iii) accuracy-based mixing.

3.4 Constant Mixing

Let us consider the case of constant mixing weights, that is $\Psi_{k,t} = \Psi_k$ for all $t = 0, 1, \ldots$ and all $k \in \{1, \ldots, K\}$. This allows us to show:

Lemma 3.6. *Let $\{\Psi_1, \ldots \Psi_K\}$ be a set of admissible mixing matrices, and let Π_{D_k} be the projection operator for averaging classifier k. Then $\sum_{k=1}^K \Psi_k \Pi_{D_k}$ is a non-expansion w.r.t. the maximum norm.*

Proof. We can show that $\sum_{k=1}^K \Psi_k \Pi_{D_k}$ is an increasing scalar shift vector field with scaling 1 in the same way as we have proven Lemma 3.4 and 3.5. Hence, from Lemma 3.2 it follows that it is a non-expansion w.r.t. the maximum norm. □

This non-expansion leads to the result:

Theorem 3.7. *Learning Classifier System Value Iteration with averaging classifiers and fixed mixing weights converges to the unique fixed point of the iteration.*

Proof. The LCS Value Iteration update for fixed mixing weights is

$$\tilde{V}_{t+1} = \sum_{k=1}^{K} \Psi_k \Pi_{D_k} T \tilde{V}_t.$$

By Lemma 3.3, T is a contraction w.r.t. $\| \cdot \|_\infty$. By Lemma 3.6, $\sum_{k=1}^K \Psi_k \Pi_{D_k}$ is a non-expansion w.r.t. the same norm. Therefore, $\sum_{k=1}^K \Psi_k \Pi_{D_k} T$ is a contraction and by Theorem 3.1 the above update converges to its unique fixed point. □

3.5 Time-variant Arbitrary Mixing

Let us now consider what happens if we change the mixing weights at every iteration. Given that the mixing weights are a function of the previous overall value approximation, would it be possible to use an arbitrary function and still get a contraction? We will again deal with this question by investigating if the LCS function approximation alone gives a non-expansion w.r.t. the maximum norm, which implies convergence of its use together with Value Iteration.

Let us consider a simple example with 2 classifiers, a state space $S = \{1, 2\}$ and uniform sampling. The first classifier matches all states, and the second classifier only matches the second state, that is $S_1 = \{1, 2\}$ and $S_2 = \{2\}$. Let the two vectors to approximate be $V = (0, 1)'$ and $\bar{V} = (2, 4)'$. Due to their averaging nature, the first classifier will give a value of $\frac{1}{2}$ for V, and a value of 3 for $\bar{V}$. The second classifier matches the values of states 2 and will therefore give 1 for V, and 4 for $\bar{V}$. As for state 2 we are mixing the approximations of both classifiers, its overall approximation $\tilde{V}(2)$ will be in the range $[0, 1]$ for V, and in the range $[2, 4]$ for $\bar{V}$, depending on the mixing weights. Note that $\|V - \bar{V}\|_\infty = 3$. As we can chose arbitrary mixing weights, let us fix the approximation of $\bar{V}(2)$ at 4. We can now observe that the difference between the approximations for $V(2)$ and $\bar{V}(2)$ is in the range $[3, 4]$ depending on the mixing weights for the approximation of V. Hence, it might be larger than $\|V - \bar{V}\|_\infty$ and therefore might violate our non-expansion property. This demonstrates that we cannot guarantee non-expansion of the LCS function approximation for arbitrary mixing weights.

Consequently, the choice of function that determines the mixing weights is significant for the contraction property of LCS Value Iteration. How does it have to be formed such that we can guarantee non-expansion? In the previous example we have put full weight on the second classifier to approximate $\bar{V}(2)$ but not to approximate $V(2)$. We assume that having some schema that weights the same classifiers in similar ways will let us show non-expansion of the function approximation. The following accuracy-based mixing is such a schema.

3.6 Accuracy-based Mixing

Let us introduce operator $\mathcal{C}$ that describes overall approximation given accuracy-based mixing. As described before, the mixing weights are based on the matching classifiers' approximation errors $\varepsilon_k : \mathbb{R}^N \to \mathbb{R}^+$, which are for classifier k given by

$$\varepsilon_k(V) = \mathrm{Tr}(D_k)^{-1} \|TV - \Pi_{D_k} TV\|^2_{D_k}.$$

The mixing weights $\psi_k : S \times \mathbb{R}^N \to [0, 1]$ are determined according to Eq. (2), and are as a function of the state and the current value function estimate given by

$$\psi_k(i, V) = \frac{I_{S_k}(i)\varepsilon_k(V)^{-\nu}}{\sum_{p=1}^K I_{S_p}(i)\varepsilon_p(V)^{-\nu}}.$$

We will write $\mathcal{C}V$ for applying this mixing strategy to a set of averaging classifiers that approximate the vector V, given by

$$(\mathcal{C}V)(i) = \sum_{k=1}^{K} \psi_k(i, V)(\Pi_{D_k} V)(i) \qquad i = 1, \ldots, N.$$

To show that the vector field $\mathcal{C} : \mathbb{R}^N \to \mathbb{R}^N$ forms a non-expansion, we will proceed as before by investigating if it is an increasing scalar shift vector field. Let us first prove the following:

Lemma 3.8. *For any vector $V \in \mathbb{R}^N$, scalar $m \in \mathbb{R}$, and vector $e \in \mathbb{R}^N$ given by $e = (1, \ldots, 1)'$ we have*

$$T(V + me) - \Pi_{D_k} T(V + me) = TV - \Pi_{D_k} TV, \quad k = 1, \ldots, K.$$

Proof. From Lemma 3.3 we know that T describes a scalar shift vector field with scaling γ. Hence we can write

$$T(V + me) - \Pi_{D_k} T(V + me) = TV - \Pi_{D_k} TV + \gamma(I - \Pi_{D_k})me.$$

Additionally, by Lemma 3.5, Π_{D_k} is a scalar shift vector field with scaling 1, and $\Pi_{D_k}(0e) = 0e$. Hence,

$$(I - \Pi_{D_k})me = me - \Pi_{D_k}(0e + me) = me - me = 0. \quad \square$$

As $TV - \Pi_{D_k} TV$ determines the classifier error, by above lemma that error is invariant under scalar shift of V, and subsequently so are the mixing weights. Hence, given that the relative differences between the values of the states that the classifier matches are correct, the error approximation is also correct. We hypothesise that therefore we can get good approximate error estimates even before the final value function is known.

We can use the invariance of the mixing weights under scalar shift to show the following:

Lemma 3.9. *The vector field given by $\mathcal{C}$ is a scalar shift vector field with scaling 1.*

Proof. By Lemma 3.8, $\varepsilon_k(V+me) = \varepsilon_k(V)$, where $V \in \mathbb{R}^N$ is any vector, $m \in \mathbb{R}$ is a scalar, and $e \in \mathbb{R}^N$ is the vector $e = (1, \ldots, 1)'$. Hence, the same can be said for the mixing weight function ψ_k, that is for any state $i \in S$, $\psi_k(i, V + me) = \psi_k(i, V)$. Lemma 3.5 shows that $\Pi_{D_k}(V + me) = \Pi_{D_k} V + me$. Hence, if we expand for $\mathcal{C}(V+me)$ and observe that $\sum_{k=1}^{K} \psi_k(i, V)m = m$ we can see that $\mathcal{C}(V + me) = \mathcal{C}V + me$. □

Having established the scalar shift property of $\mathcal{C}$, we will now investigate if it is increasing. From the definition of the mixing weights ψ_k we can see that for $\nu = 0$, ψ_k is independent of the current value function estimate and therefore time-invariant. Hence, we can apply Lemma 3.6 to show that $\mathcal{C}$ is increasing. However, as is discussed in [3], $\nu = 0$ is possibly the worst setting for this parameter. Thus, we are more interested in the properties of $\mathcal{C}$ for $\nu > 0$.

It is well known that a differentiable continuous function of a single variable is increasing if and only if its first gradient is non-negative (e.g. [1, Ch. 11.2]). As derived in [5], the same principle applies to vector fields with a non-negative Jacobian. For $\mathcal{C}$ the components of the Jacobian are given by[10]

$$\begin{aligned} \frac{\partial \mathcal{C}_i V}{\partial V(l)} &= \sum_{k=1}^{K} \psi_k(i, V) \frac{I_{S_k}(l)\pi(l)}{\mathrm{Tr}(D_k)} \\ &\quad \times \left(1 + 2\nu\varepsilon_k(V)^{-1}(V(l) - V_k)(\mathcal{C}_i V - V_k)\right), \end{aligned}$$

where $\mathcal{C}_i V = (\mathcal{C}V)(i)$ is the ith component of the result of $\mathcal{C}V$, and V_k stands for any component of $\Pi_{D_k} V$. Given that the above gives a non-negative result for all $i = 1, \ldots, N$ and $l = 1, \ldots, K$, the vector field described by $\mathcal{C}$ is increasing.

At present, our analysis has not identified whether the components of the Jacobian are non-negative and this part of the investigation is future work. Discovering that some of the components are negative does not imply that LCS Value Iteration diverges, as the non-negativity of the components is only a sufficient but not a necessary condition. Neither can we reject convergence if $\mathcal{C}$ is found to be non-increasing, as an increasing vector field is sufficient to apply Lemma 3.2, but not proven to be necessary for convergence.

As we have not yet been able to produce a proof that the vector field given by $\mathcal{C}$ is increasing, but neither were we able to find examples where it violates that property, we will state it as a conjecture, pending further investigation.

Conjecture 3.10. *The vector field given by $\mathcal{C}$ is increasing.*

This leads to the following result:

[10] For the derivation of the Jacobian of $\mathcal{C}$ see [5]

Theorem 3.11. *If Conjecture 3.10 holds, then LCS Value Iteration with accuracy-based mixing converges to its fixed point $\tilde{V}^* = \mathcal{C}T\tilde{V}^*$.*

Proof. As by Lemma 3.9, $\mathcal{C}$ is a scalar shift vector field with scaling 1, and by Conjecture 3.10 it is increasing, it forms by Lemma 3.2 a non-expansion w.r.t. the maximum norm. Therefore, by Lemma 3.3, $\mathcal{C}T$ forms a contraction, and by Theorem 3.1 the LCS Value Iteration $\tilde{V}_{t+1} = \mathcal{C}T\tilde{V}_t$ converges to its unique fixed point. □

4 CONCLUSION

We have described the LCS Value Iteration algorithm and have given a proof of its convergence for constant mixing and accuracy-based mixing, where the latter is based on a conjecture about LCS function approximation. Furthermore, we have shown that the choice of the mixing function is significant for the convergence of the algorithm.

Convergence of LCS Value Iteration is an important property, as it is the first step towards studying the stability of using Q-Learning in LCS. Even though we are currently only dealing with constant populations of classifiers, showing convergence to a population-dependent fixed point allows us to use this work even when we are changing the population while we are performing the iteration. In that case, the fixed point would change, but every iteration after that change would bring us closer to the new fixed point. Naturally, we also have to consider that the new population depends on the previous value function estimate, and analysing this interaction is a topic of future research.

Having guaranteed convergence to a unique solution is a strong property that makes classifier systems better candidates for real-world application, such as, for example, optimal control. Hence, following this track of research for LCS will be fruitful for a wide range of applications that have not previously been considered before due to the lack of theoretical guarantees.

Acknowledgements Thanks to Jonty Needham for being patient enough to listen to a large number of naïve math questions, and to answer some of them in an understandable, and sometimes not-so-understandable way.

References

[1] Howard Anton, *Calculus*, John Wiley & Sons, New York, 5th edn., 1995.
[2] Dimitri P. Bertsekas and John N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, MA, 1996.
[3] Jan Drugowitsch and Alwyn M. Barry, 'A Formal Framework and Extensions for Function Approximation in Learning Classifier Systems', Technical Report CSBU-2006-01, Dept. Computer Science, University of Bath, (January 2006). ISSN 1740-9497.
[4] Jan Drugowitsch and Alwyn M. Barry, 'A Formal Framework for Reinforcement Learning with Function Approximation in Learning Classifier Systems', Technical Report CSBU-2006-02, Dept. Computer Science, University of Bath, (January 2006). ISSN 1740-9497.
[5] Jan Drugowitsch and Alwyn M. Barry, 'Towards Convergence of Learning Classifier Systems Value Iteration', Technical Report CSBU-2006-03, Dept. Computer Science, University of Bath, (April 2006). ISSN 1740-9497.
[6] Geoffrey J. Gordon, 'Stable Function Approximation in Dynamic Programming', in *Proceedings of the Twelfth International Conference on Machine Learning*, eds., Armand Prieditis and Stuart Russell, pp. 261–268, San Francisco, CA, USA, (1995). Morgan Kaufmann.
[7] W. A. Sutherland, *Introduction to metric and topological spaces*, Clarendon Press, Oxford, UK, 1975.
[8] Stewart W. Wilson, 'Classifiers that Approximate Functions', *Neural Computing*, **1**(2-3), 211–234, (2002).